Honorin Pouteau
Student ID: H00382694

Web Browser Project Report

1. Introduction

In this report is to describe the development process of the first coursework of the industrial programming course.
The objective of this coursework was to create a simple web browser using C# with the knowledge acquired during the course.
This report will be firstly explaining the requirements necessary to the completion of the Web browser and these tasks have been successfully done before the deadline.
Next it will be the design consideration made during the implementation of the GUI or Graphical user Interface.
The report also includes a user guide to help the user navigate on the application.
The Developer guide is explaining the technical decision made to complete the project like the platform and library used or the code architecture employed and the reason. This part will help new developers understand and potentially expand the functionality of the Web Browser.
After that we will explain some principal test cases used to prove that the application performs well even in unexpected conditions.
Before concluding this document will reflect on the development process, principally about the technology used, the mistakes made along the way.

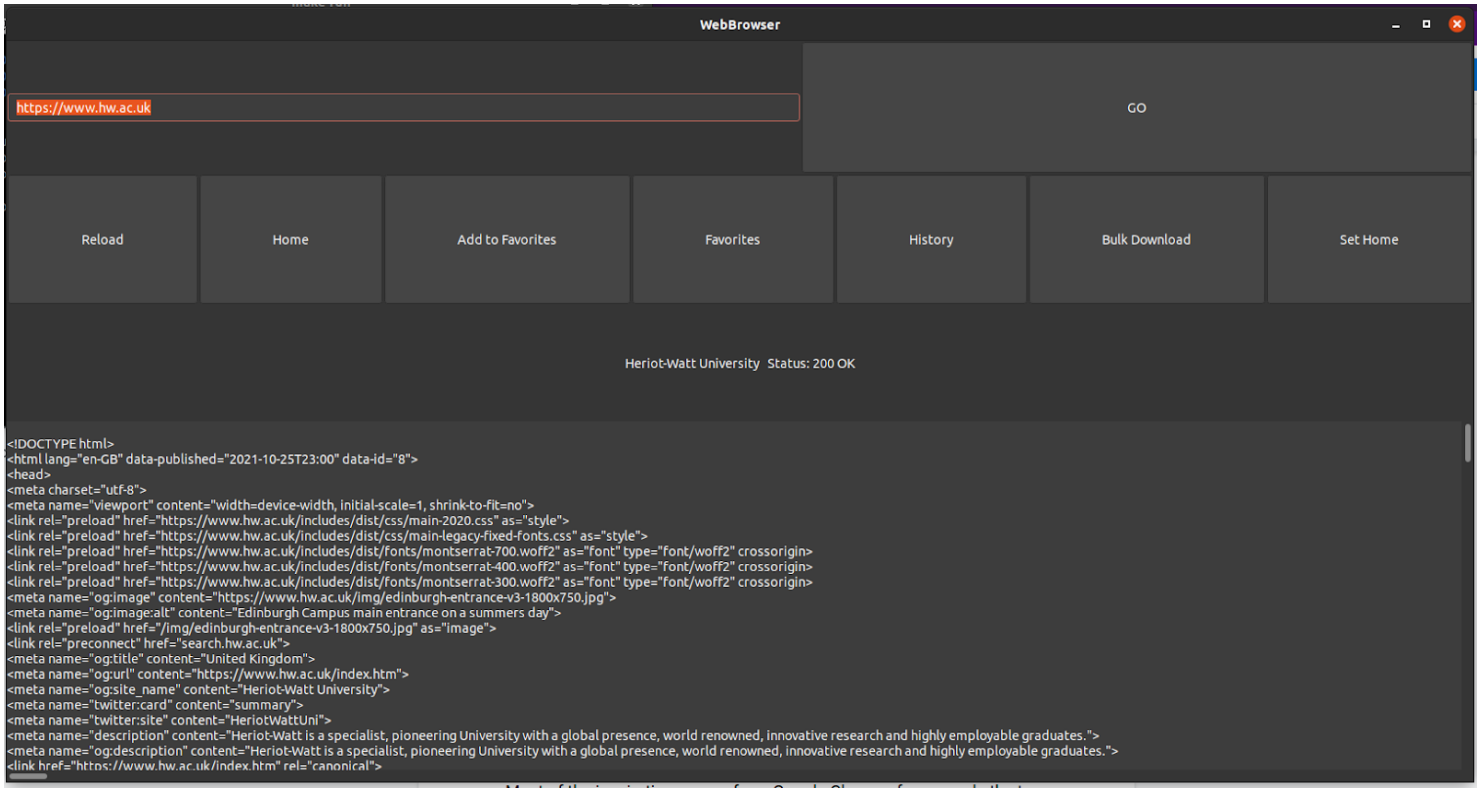
2. Requirements' checklist

Requirements	Delivered
Display	
Sending HTTP Request messages to URL typed by the user	Yes
Receiving and display the response messages on the interface (HTML)	Yes
Display HTTP status code and corresponding error messages	Yes
Display the title of the web page	Yes
Web page can be reloaded with new HTTP request	Yes
Home page	
The browser has a home page URL loaded on browser start-up (Initialised on https://www.hw.ac.uk/)	Yes
The user can edit the home page	Yes
Favourites	
The user can add and remove a URI to a list of favourites	Yes
The user can request a favourite URI by clicking on it in his favourites URL list	Yes
The user can associate a name with a favourite and modify it	Yes
The favourites list is loaded to the browser on start-up	Yes
History	

The browser maintains a list of the URL requested by the user	Yes
The user can request a previously requested URL by clicking on it in a history list	Yes
The history list is loaded to the browser on start-up	Yes
Bulk download	
The user can specify a file name containing a one URL per line which will be requested by the browser	Yes
For each URL requested the browser display one line containing the status code, number of bytes and the URL of the request	Yes
GUI	
The user can use a GUI to perform the action described above	Yes
The user can use different menus (containing buttons) to interact easily with the browser	Yes
The user can use some shortcut to navigate the browser	Yes

3. Design Consideration

The Web Browser was originally intended to be done on the Linux system, with the Mono infrastructure for C# and .Net and the GUI was to be done with Gtk#.
The project was compiled in the terminal using a makefile.
This configuration was chosen because of software accessibility and a certain competence with it.
But later during development it was discovered that the Gtk# library was very poorly documented and was greatly limiting the choice of interface design possible for the project. It also needed the installation of gtk# in the machine trying to use it and it wasn't possible to install on the machines in the labs.
The design at the end of the Linux development was functional but the GUI was lacking as you can see below.



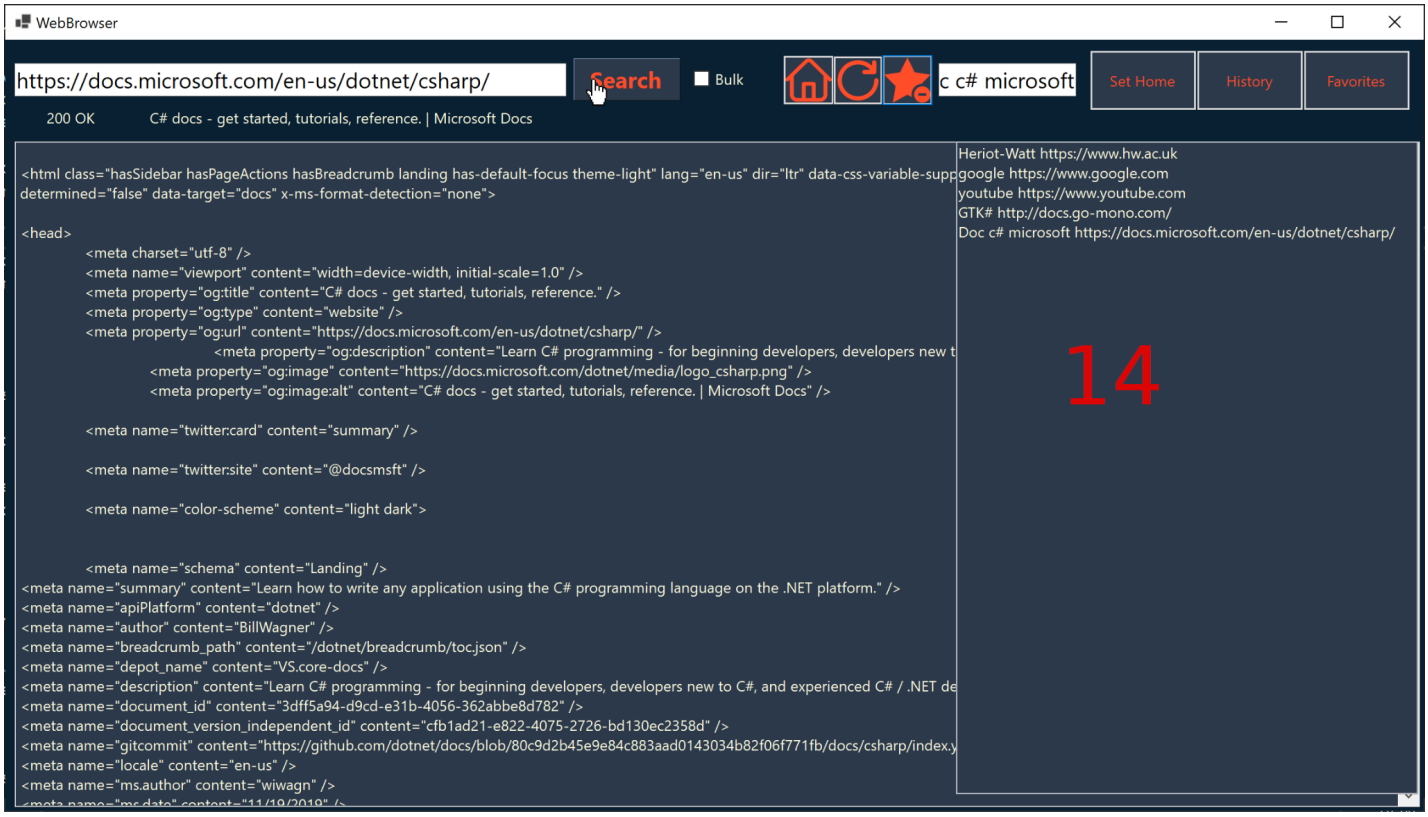
The code and executable for this still exist on the "mono/gtk#" branch on the Gitlab repository.

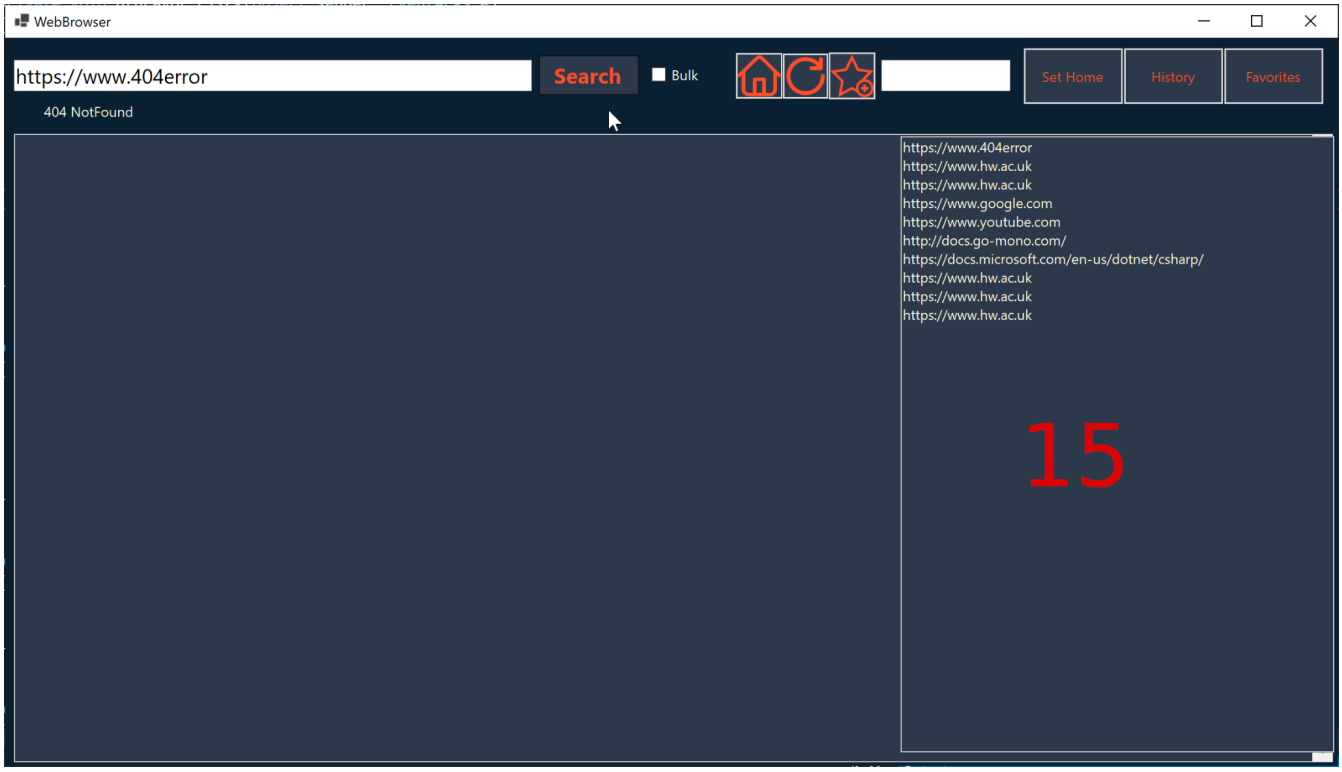
The project was moved to a windows system using Visual Studio for the edition and compilation of the solution and Windows form for the GUI. Windows forms was very easy intuitive and easy to handle to create a more good-looking web browser as opposed to Gtk#.

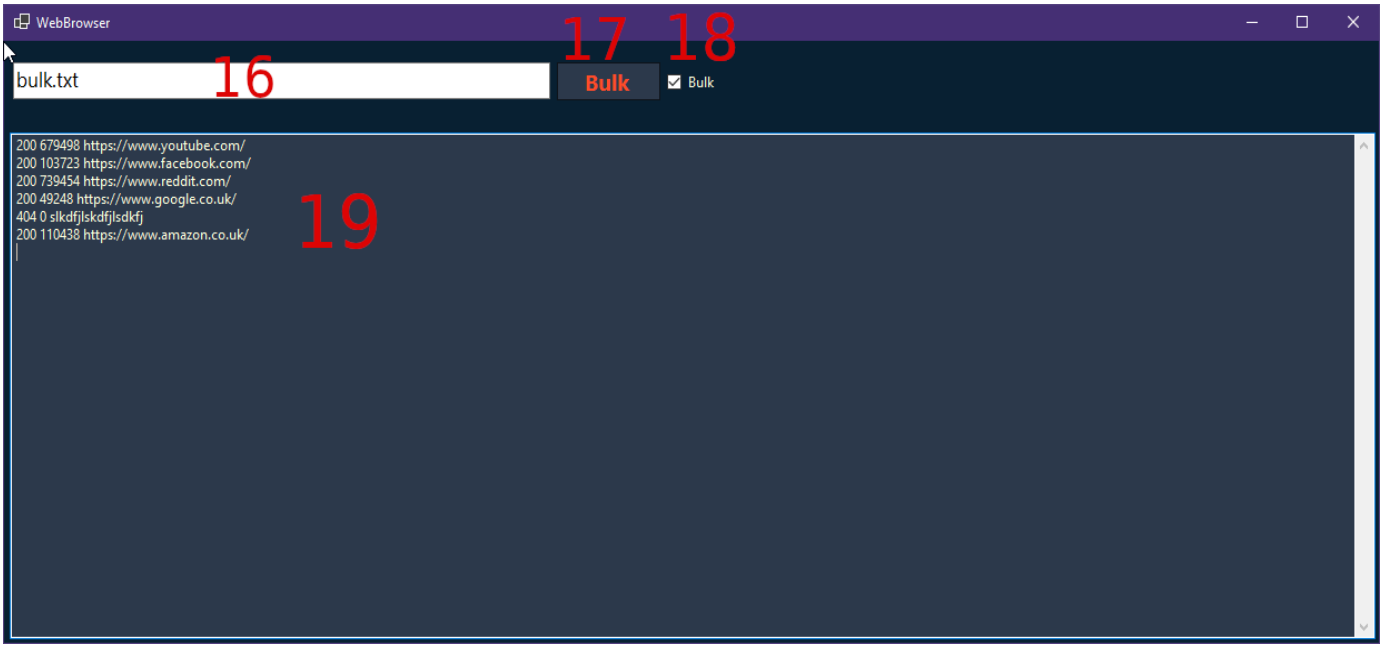
The GUI(Graphic User Interface) design of the web browser is largely inspired by the most commonly used Web Browser already existing in an attempt to not lose the user who is used to these softwares.
Most of the inspiration comes from Google Chrome, for example the top bar design uses a very similar button pattern and look.
A dark theme was chosen for the application in order for it to become less aggressive to the eyes of the users.

4. User Guide


WebBrowser
https://www.hw.ac.uk
1
Search
4
Bulk
5
6
7
8
9
10
11
12
Set Home
History
Favorites
2
200 OK
3
Heriot-Watt University
13


WebBrowser
https://docs.microsoft.com/en-us/dotnet/csharp/
Search
Bulk
c c# microsoft
Set Home
History
Favorites
200 OK
C# docs - get started, tutorials, reference. | Microsoft Docs
Heriot-Watt https://www.hw.ac.uk
google https://www.google.com
youtube https://www.youtube.com
GTK# http://docs.go-mono.com/
Doc c# microsoft https://docs.microsoft.com/en-us/dotnet/csharp/
14


WebBrowser
https://www.404error
Search
Bulk
Set Home
History
Favorites
404 NotFound
https://www.404error
https://www.hw.ac.uk
https://www.hw.ac.uk
https://www.google.com
https://www.youtube.com
http://docs.go-mono.com/
https://docs.microsoft.com/en-us/dotnet/csharp/
https://www.hw.ac.uk
https://www.hw.ac.uk
https://www.hw.ac.uk
15


WebBrowser
17
18
bulk.txt
16
Bulk
Bulk
200 679498 https://www.youtube.com/
200 103723 https://www.facebook.com/
200 739454 https://www.reddit.com/
200 49248 https://www.google.co.uk/
404 0 slkdfjlskdfjlsdkfj
200 110438 https://www.amazon.co.uk/
19

Main menu	
1	Search Bar to input website URI address
2	Status code and message
3	Title of the page when applicable
4	Search button used to request URI in search bar (shortcut key enter when in the search bar)
5	Activate bulk download mode
6	Home button to go to the home URI (shortcut key home)
7	Reload button used to reload page (shortcut key f5)
8	Favorite button used add favorite (+) or remove them (-)
9	Text input used to attach name to favorite
10	Set Home button used to change the home URI
11	History List button used to open the history menu
12	Library List button used to open the library menu
13	Main body where the html of the page is displayed
History and Favorites menu	
14	Main body where the html of the page is displayed History and Favorites menu
15	Favorite menu with names and URI. Clicking an element make a request to the element URI
Bulk download menu	
16	Test input for the file path to do the bulk download
17	Bulk button will validate the bulk file and do the download of the bulk(same shortcut key as the search button, enter)
18	Quit Bulk mode and go back to the browser
19	Body containing the result of the bulk download

A demo screencast is available in the projects file or in the link below.
https://heriotwatt-my.sharepoint.com/:v:/g/personal/hp2017_hw_ac_uk/Ee9_7zJ62pVFtTc-8YiVfDEBEfPfuI4mzMKxs2ETGw8Q-Q?e=eHZyPO

5. Developer Guide

The Web Browser is now constituted of three main classes :

- The Request class
- The Database class
- The WebBrowser class

The first two were recovered from the previous iteration of the project on Linux.
The Request class objective is to make a request to all URI in the file(get and bulk download) using the HttpClient library which was chosen for its wide use and its simplicity.
The Database class has to oversee all operations on the database which contain information about the home page, favorites and history that have to be stored between the application uses.
For these procedures Linq and Linq to Xml are used to aggregate the data and create a “database.xml” file.
This method was used because it is very easy to read and write from and to the database.
The Web Browser class was the intermediate between the two first class and Windows Form, it contains all the logic of the interface and uses Request and Database to display the correct information.

6. Testing

For this project we tested a lot of possible error cases.
Here a few examples :

- Request to non existing URI
- Different Status code on request
- No name associated with the favorite
- Database file not existing
- Bulk file doesn't exist
- Browser menu and shortcut key not accessible in bulk and vice-versa

None the possible error case that were tested cause a crash in the project

7. Reflections on programming language and implementation

Looking back on the project it was a bad idea to do the application on Linux using gtk# but this constraint forced the utilisation of more interesting language features of the C# like delegates, interface or generic type that didn't need to be used on windows with visual studio and Windows Forms.

8. Conclusions

To conclude this report, this project made me learn a lot about the C# language and the different features that can be used in it as well as developing software on windows.
I'm happy that all the requirements are delivered but even though It made me learn a lot about C# I lost a lot of time developing the project on Linux and Gtk#, I could have used this time to add more functionality to the project.